Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Andrej Premrn

Spletna aplikacija za ocenjevanje ponudnikov prehrane na študentske bone

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Viljan Mahnič

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Izdelajte spletno aplikacijo, ki bo uporabnikom omogočala iskanje in ocenjevanje ponudnikov prehrane na študentske bone. Pri tem posvetite posebno pozornost izboru ustreznih orodij, aplikacijo pa načrtujte v skladu z načeli odzivne zasnove spletnih strani (angl. responsive web design), tako da jo bo moč uporabljati na različnih napravah: stacionarnih računalnikih, tabličnih računalnikih in pametnih telefonih.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Andrej Premrn, z vpisno številko **63110336**, sem avtor diplomskega dela z naslovom:

Spletna aplikacija za ocenjevanje ponudnikov prehrane na študentske bone

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Viljana Mahniča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 16. septembra 2015 Podpis avtorja:

Zahvaljujem se družini in vsem svojim bližnjim, ki so me spodbujali in podpirali tekom študija. Zahvalil bi se rad tudi mentorju, izr. prof. dr. Viljanu Mahniču za vse nasvete in pomoč pri izdelavi diplomskega dela. Posebej bi se rad zahvalil še Aneju Budihni, Žanu Hočevar Pegancu in Klemenu Klančiču, ki so mi posodili prenosne naprave za testiranje aplikacije.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
MVC	model view controller	model pogled kontroler
URL	uniform resource locator	enolični krajevnik vira
HTML	hypertext markup language	označevalni jezik za oblikovanje večpredstavnostnih dokumentov
DOM	document object model	model za predstavitev objektov dokumenta
CSS	cascading style sheet	stilna predloga
DRY	don't repeat yourself	ne ponavljaj se
API	application programming interface	programski vmesnik
RWD	responsive web design	odzivna zasnova spletne strani
NPM	node package manager	upravljalec node paketov
JSON	JavaScript object notation	notacija JavaScript objektov
URI	uniform resource identifier	enotni označevalnik vira

Povzetek

V diplomskem delu je predstavljen razvoj spletne aplikacije za ocenjevanje ponudnikov prehrane na študentske bone. Opisane so razvojne tehnologije in orodja, zahtevane funkcionalnosti, razvojni proces in končna funkcionalnost aplikacije. Osredotočili smo se tudi na odzivnost spletne strani, da lahko aplikacijo uporabljamo tudi na pametnih prenosnih napravah. Pri razvoju smo se osredotočili na platformo Node.js, obogateno z ogrodjem Express.js in druge spletne tehnologije, kot so: Jade, Bootstrap in jQuery. Podatke smo pridobivali s tehniko grabljenja podatkov, ki smo jo realizirali s programskim jezikom Python. Za hranjenje podatkov smo uporabili podatkovno bazo MySQL. Funkcionalnost aplikacije obsega iskanje ponudnikov študentske prehrane, njihovo anonimno ocenjevanje in komentiranje ter iskanje ponudnikov v bližini trenutne lokacije, kar je ena izmed glavnih funkcionalnosti.

Ključne besede: spletna aplikacija, Node.js, odzivnost spletne strani, subvencionirana študentska prehrana.

Abstract

The thesis describes the development of a web application for rating subsidized students meals providers. It describes development technologies and tools used, required functionality, development process and final functionality of the application. We also focused on responsive web design, so that the application can also be used on smart handheld devices. While developing we focused on Node.js platform enriched with Express.js framework and other web technologies such as: Jade, Bootstrap and jQuery. Data about providers was acquired with web scraping technique, which we implemented with the programming language Python. For data storage we used a MySQL database. The application functionality consists of searching for providers, their anonymous rating and commenting and also searching for providers near your current location, which is one of the main functionalities.

Keywords: web application, Node.js, responsive web design, subsidized students meals.

Poglavje 1

Uvod

Večina študentov vsaj enkrat dnevno koristi možnost subvencionirane prehrane. Ob tem se poraja vprašanje katerega ponudnika izbrati. Navadno se študentje odločajo glede na pretekle izkušnje, predloge prijateljev in oddaljenost. Na spletu lahko trenutno najdemo nekaj blogov[1], kjer avtorji opisujejo svoje izkušnje pri ponudnikih subvencionirane prehrane, spletna stran Odpiralni časi[2] pa obiskovalcem omogoča ocenjevanje in komentiranje restavracij, vendar je po našem mnenju preveč splošna in uporabnika zasuje s preveč informacijami.

Pojavila se je ideja, da izdelamo spletno aplikacijo, s katero bi lahko študentje delili svoja mnenja in ocene še z drugimi. Poleg komentiranja in ocenjevanja, bi lahko uporabniki še iskali ponudnike v bližini trenutne ali poljubne lokacije. Naša aplikacija bi tako lahko zmanjšala število primerov, ko je študent obravnavan kot drugorazredni gost, kar je bila glavna motivacija pri realizaciji ideje. Ker živimo v dobi pametnih prenosnih naprav, smo spletno stran tudi optimizirali za prikaz na manjših zaslonih.

V prvem delu diplomskega dela opredelimo zahteve in funkcionalnosti aplikacije. V drugem delu predstavimo tehnologije in orodja, ki smo jih upora-

[1] https://restavracijeljubljana.wordpress.com in http://nabonapetit.blogspot.si

[2] http://odpiralnicasi.com

bljali pri realizaciji. V nadaljevanju se osredotočimo na opis poteka dela in med drugim še opišemo končno aplikacijo. Podrobneje opišemo, kako smo vzpostavili delovno okolje, pridobili podatke o ponudnikih in izdelali samo aplikacijo. V sklepu naštejemo nekaj možnih izboljšav in omenimo nekaj ugotovitev.

Poglavje 2

Opredelitev zahtev in funkcionalnosti

Pri snovanju ideje in zahtev aplikacije smo se osredotočili predvsem na uporabnost in uporabniško izkušnjo. Razmišljali smo tudi o tem, kdo je končni uporabnik in kako bo uporabljal aplikacijo. Končni uporabnik je študent, ki koristi študentske bone (v nadaljevanju besedila se na študenta sklicujemo z besedo (navaden) uporabnik). Slednjemu moramo omogočiti uporabo aplikacije tako doma, kot v restavraciji ali kje drugje. Zato smo se odločili, da je potrebno uporabiti odzivno zasnovo spletne strani (RWD), da bo aplikacija pravilno delovala in se prikazovala na računalniku in prenosnih napravah kot so pametni telefoni in tablice. Aplikacija naj bi bila karseda minimalistična v smislu dizajna in funkcionalnosti, kar bi pomenilo, da uporabnika ne zasuvamo z nepotrebnimi informacijami in, da aplikacija opravlja le nekaj ključnih nalog. Poleg navadnega uporabnika imamo še skrbnika spletne aplikacije. Njegova glavna naloga je pridobivanje in posodabljanje podatkov o ponudnikih.

2.1 Navaden uporabnik

Navaden uporabnik je vsak obiskovalec spletne strani, ki je v večini primerov študent. Registracija uporabniškega računa naj ne bi bila potrebna, saj to po našem mnenju kvari uporabniško izkušnjo in nenazadnje tudi hočemo, da obiskovalci, ki komentirajo in ocenjujejo ostanejo anonimni.

2.1.1 Ocenjevanje in komentiranje ponudikov

Uporabnik lahko oceni ponudnika z oceno med 1 in 5 in odda svoje mnenje v komentarju dolgemu največ 1000 znakov.

2.1.2 Pregled nekaterih informacij, ocen in komentarjev o ponudnikih

Prikazani naj bi bili podatki o imenu in naslovu ponudnika, povprečna ocena in cena doplačila obroka. Dodatno naj bo prikazana tudi povezava profila ponudnika na uradno stran študentske prehrane [12], kjer lahko uporabnik pridobi še ostale informacije, ki jih aplikacija ne izpisuje. Na podstrani ponudnika naj se izpišejo vsi oddani komentarji, ki jih je možno urejati.

2.1.3 Iskanje in razvrščanje ponudnikov

Uporabnik ima možnost iskanja glede na ime, naslov in oceno. Rezultate iskanja naj bi bilo možno urediti po abecedi ali oceni.

2.1.4 Iskanje ponudnikov v bližini trenutne lokacije

Na zemljevidu se v obsegu (določi ga uporabnik) od trenutne lokacije, ki se privzeto določi sama, poljubno pa jo lahko določi uporabnik sam, prikažejo

lokacije ponudnikov in njihova ocena. Uporabnik lahko s klikom na označbo dostopa do podstrani ponudnika.

2.2 Skrbnik spletne aplikacije

Odločili smo se, da skrbnik spletne aplikacije ne bo imel namenskega računa na spletni strani, ampak bo vse vzdrževanje potekalo preko podatkovne baze, npr. z orodjem phpMyAdmin. Podatke skrbnik pridobi in posodablja z uporabo Python skript.

2.2.1 Pridobivanje podatkov o ponudnikih

Z uporabo tehnike grabljenja podatkov naj se pridobijo željeni podatki z uradne strani o študentski prehrani. Podatki naj se uredijo v SQL stavke in zapišejo v datoteko, ki je primerna za uvoz v podatkovno bazo.

2.2.2 Posodabljanje podatkovne baze z novimi podatki

Na novo pridobljeni podatki o ponudnikih naj se primerjajo z zapisi v podatkovni bazi, ki naj se po potrebi posodobijo. Novi ponudniki naj se dodajo v bazo, tisti, ki so izgubili možnost nudenja subvencionirane prehrane, naj se v polju v bazi ustrezno označijo. Taki ponudniki naj se izločijo iz prikaza na strani.

Poglavje 3

Pregled tehnologij in orodij

Za razvoj naše spletne aplikacije smo se odločili uporabiti nekaj novejših tehnologij. Prvi in najpomembnejši razlog za to odločitev je bil ta, da se seznanimo z novimi tehnologijami in hitrim razvojem spletnih aplikacij. Drugi razlog je bil ta, kako in ali lahko naše, že obstoječe znanje iz razvoja spletnih aplikacij prenesemo in apliciramo na novejše tehnologije in orodja, saj se z veliko večino uporabljenih programskih jezikov in tehnologij še nikoli prej nismo srečali.

3.1 Glavne razvojne tehnologije in orodja

Na strežniku (back end) smo uporabili Express.js. Za obličje spletne strani (front end) skrbi kombinacija Bootstrapa, ki skrbi predvsem za stilsko predlogo, jQueryja, ki skrbi za dinamičnost spletne strani in Jadea, ki generira naše HTML dokumente. Pri izbiri podatkovne baze smo izbirali med NoSQL [22] podatkovno bazo MongoDB [19], ki se navadno uporablja v navezi z Express.js in bolj tradicionalno relacijsko bazo MySQL. Ker ne pričakujemo veliko pisanj in potrebe po skalabilnosti, smo izbrali slednjo.

Podatke o ponudnikih pridobivamo z uradne spletne strani[1]. Za ta namen smo uporabili programski jezik Python, ki s knjižnicami zelo poenostavi grabljenje podatkov (angl. web scraping).

3.1.1 Node.js in Express.js

Node.js [21] je odprtokodno okolje za poganjanje strežniških in spletnih aplikacij. Aplikacije, ki jih poganja, so napisane v jeziku JavaScript. Node.js omogoča, da se aplikacija obnaša kot spletni strežnik brez prav za to namenjene programske opreme kot, sta npr. Apache in Nginx.

Express.js [7] je minimalistično ogrodje, zgrajeno na platformi Node.js. Odraža ga predvsem robusten nabor funkcionalnosti in možnost organizacije spletne aplikacije v MVC arhitekturo. Ena izmed najboljših funkcionalnosti je usmerjevalni mehanizem (routing), ki omogoča, da z internetnim naslovom kontroliramo stanje spletne aplikacije.

3.1.2 jQuery

JQuery [14] je bogata, ampak hkrati majhna knjižnica za JavaScript. Njen glavni namen je, da poenostavi kodo, ki je v JavaSriptu zapletena in dolga. Omogoča enostavno HTML/DOM manipulacijo, CSS [6] manipulacijo, odziv na dogodke, animacijo in poenostavljene AJAX [2] klice. Če potrebujemo kakšno funkcionalnost, ki jo jQuery v osnovi ne podpira, jo lahko dodamo z raznimi vtičniki.

3.1.3 MySQL

MySQL [20] je odprtokodni sistem za upravljanje relacijskih podatkovnih baz. Je majhen hiter in enostaven za uporabno. Podatki v podatkovni bazi

[1] https://www.studentska-prehrana.si

MySQL so shranjeni v tabelah. Tabele lahko preko primarnih in tujih ključev povežemo in tako tvorimo relacije med nijimi. Podatke iz baze pridobimo tako, da z jezikom SQL [25] tvorimo ustrezne poizvedbe.

3.1.4 Jade

Jade [13] je sistem za generiranje HTML [11] predlog. Koda je dinamična in zelo podobna HTMLju, s tem, da je veliko poudarka na DRY pristopu programiranja, ki ga omogočajo t.i. mixini [18], ki so v primeru Jadeja kot nekakšne metode, ki glede na podane parametre generirajo nek element. Jade je privzet sistem za generiranje predlog v Node.js. V principu deluje tako, da v kodi na strani strežnika pridobimo ali generiramo željene podatke, ki jih nato z Jadeom ustrezno umestimo v predlogo. Predloga se nato na strani strežnika prevede in kot rezultat dobimo HTML, ki se prikaže uporabniku.

3.1.5 Bootstrap

Bootstrap [3] je ogrodje podjetja Twitter, ki zajema HTML, CSS in JavaScript komponente. Namenjen je hitremu in enostavnemu razvijanju obličja spletnih strani in aplikacij. Z njegovo pomočjo našim HTML elementom dodajamo različne, predpripravljene razredne značke, ki jih Bootstrap prepozna in ta element obogati s stilom in funkcionalnostmi iz svojih datotek.

3.1.6 Python

Python [24] je večnamenski, visoko nivojski programski jezik. Podpira funkcijsko, proceduralno in objektno programiranje. Podatkovni tipi so popolnoma dinamični in delo s pomnilnikom je avtomatizirano. Njegova sintaksa in obsežna standardna knjižnica omogočata, da kodo napišemo na krajši in lepši način kot v nekaterih drugih višje nivojskih jezikih.

3.2 Ostala orodja, ogrodja in knjižnice

Pri izdelavi spletne aplikacije smo si pomagali še z:

- urejevalnikom besedil Sublime Text [26],
- stilsko predlogo spletne strani Bootswatch Sandstone [5],
- Google Maps APIjem [9] za prikaz ponudnikov na zemljevidu,
- phpMyAdmin [23] za pregledovanje in nadzor podatkovne baze,
- knjižnico Bootstrap Rating [4] za implementacijo izbire ocene ponudnika,
- asinhronske klice AJAX [2], da ne po nepotrebnem osvežujemo spletne strani,
- knjižnico GeoPy [8] za Python, s katero smo določali geografsko lokacijo ponudnikov,
- knjižnico Lxml [17] za Python, s katero smo si pomagali pri grabljenju podatkov o ponudnikih,
- poizvedovalni jezik XPath [27], s katerim smo izluščili podatke o ponudnikih,
- spletnimi brskalniki Chrome, Firefox in privzetim Android brskalnikom za testiranje delovanja spletne aplikacije.

Poglavje 4

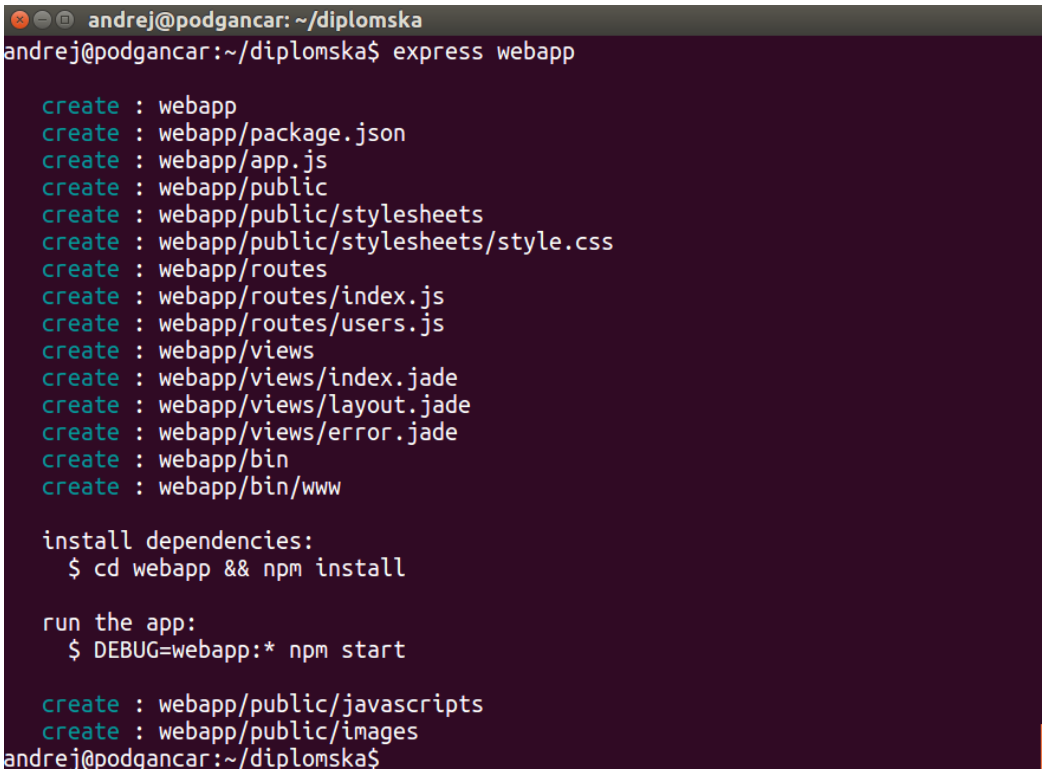
Načrt in potek dela

Razvoj je potekal na operacijskem sistemu Ubuntu 14.04 LTS in se je v grobem delil na dva dela in sicer na razvoj skript za pridobivanje podatkov in na razvoj spletne strani. Razvoj spletne strani je potekal izmenično na programiranju obličja (front-end) in kode na strežniku (back-end).

4.1 Vzpostavitev delovnega okolja in generiranje osnovne Express.js aplikacije

Začetek razvoja se je začel z namestitvijo urejevalnika besedil Sublime Text 3. Sublime je bil odlična izbira, ker podpira barvanje kode za večino programskih jezikov, ima vgrajene prijeme za hitro generiranje polj in seznamov, hiter dostop do željenega mesta v kodi in še mnoge druge funkcionalnosti. Ena izmed njih je tudi dodajanje vtičnikov, s katerim smo dodali vtičnik za barvanje Jade kode, ki ga urejevalnik sam ne podpira.

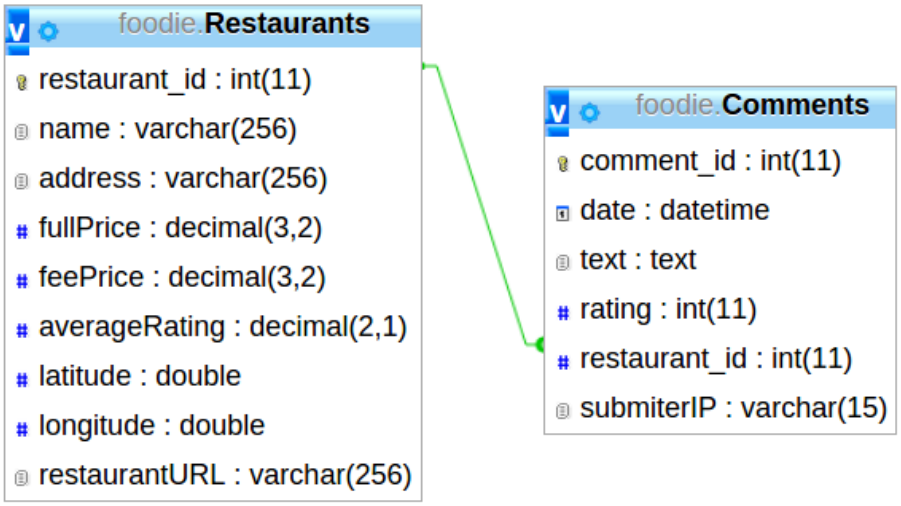
Naslednji korak je bil namestitev izvajalnega okolja Node.js. Z uradne spletne strani smo prenesli namestitveno datoteko in jo namestili. Poleg Node.js se je namestil tudi upravljalec node paketov (NPM). NPM nam omogoča,



Slika 4.1: Datotečna struktura naše spletne aplikacije.

da privzeto funkcionalnost Node.js razširjamo z dodajanjem modulov. Nato smo se s pomočjo sistemskega terminala premaknili v delovni direktorij in s pomočjo NPM in ukaza *npm install -g express-generator* namestili generator ogrodja oziroma osnovne datotečne strukture za aplikacijo Express.js. Z ukazom *express webapp* smo nato zgenerirali datotečno strukturo naše aplikacije, ki je prikazana na sliki 4.1.

Datoteka *package.json* vsebuje podatke o modulih in njihovih verzijah, kar olajša njihovo posodabljanje. V mapi *public* se nahajajo skriptne datoteke, stilske predloge in slike. Mapa *routes* vsebuje datoteke s kodo, ki skrbi za usmerjanje uporabniških zahtevkov. Mapa *views* vsebuje vse datoteke, iz katerih se generira obličje spletne strani. Datoteki *app.js* in *www* sta srce naše aplikacije. V njej se sklicujemo na vse module, povežemo usmerjevalnike, inicializiramo dostop do baze in določmo vrata ter naslov za dostop.



Slika 4.2: Struktura tabel podatkovne baze.

V našem primeru smo potrebovali še modul MySQL, ki je prirejen za platformo Node.js. V *package.json* smo dodali informacije o MySQL modulu in njegovi verziji. Namestili smo ga z ukazom *npm install*. Ta ukaz pregleda datoteko *package.json* in namesti vse potrebne module (dependecies), ki še niso nameščeni. Po namestitvi MySQL modula smo imeli vse pripravljeno za razvijanje spletne aplikacije. V terminalu smo pognali ukaz *npm start* in v brskalniku na naslovu `http://localhost:3000` dostopali do osnovne Express.js aplikacije, ki nam jo je zgeneriral generator.

4.2 Načrtovanje in izdelava podatkovne baze

Na načrtovanje podatkovne baze je veliko vplivala želja naj bo aplikacija enostavna in uporabnika ne zasuje z nepotrebnimi informacijami. V bazi imamo dve tabeli in sicer tabelo *Restaurants* in tabelo *Comments*. Relacijski model in struktura tabel je prikazan na sliki 4.2. Tabeli sta v povezavi *1:N*, kar pomeni, da ima lahko vsaka restavracija več komentarjev.

V tabeli *Restaurants* hranimo tudi lokacijo ponudnika v poljih *latitude* in *longitude*. Lokacijo smo določili že predhodno, zaradi omejitve števila poizvedb, ki jih lahko opravimo z brezplačnimi geolokatorji, kar bi povzročalo probleme, če bi določali lokacijo vsakič, ko bi uporabnik hotel prikazati ponudnike na zemljevidu, saj bi zaradi dokaj velikega števila ponudnikov hitro prekoračili dovoljeno kvoto. V polju *restaurantURL* hranimo povezavo do ponudnika na uradni strani o študentski prehrani. Uporabniku hočemo zagotoviti, da lahko hitro dostopa do informacij o ponudniku, ki jih naša aplikacija ne prikazuje.

Ker za uporabo naše aplikacije ni potrebna registracija in lahko uporabniki oddajajo komentarje anonimno, smo se odločili, da bomo v tabeli *Comments* v polju *submiterIP* hranili tudi IP naslov, s katerega je bil komentar oddan. Hramba IP naslova nam omogoča, da v primeru smetenja ali 'ponarejanja' mnenj IP naslov blokiramo, oziroma preko IP naslova odkrijemo tako obnašanje.

4.3 Pridobivanje podatkov o ponudnikih

Podatke o ponudnikih smo pridobili s tehniko grabljenja podatkov. Pri tehniki grabljenja podatkov, pridobimo DOM zgradbo spletne strani, se po njej sprehajamo in iz HTML elementov pridobivamo željene podatke, kot so atributi in vrednosti. V našem primeru smo za grabljenje in urejanje podatkov uporabili programski jezik Python in knjižnico Lxml.

Izsek kode 4.1 prikazuje uporabo knjižnice Lxml. Najprej smo z uporabo metode *get*, kateri smo podali URL naslov imenika ponudnikov, pridobili objekt spletne strani. Nato smo z uporabe metode *fromstring* iz teksta strani generirali drevo HTML objektov. V naslednjem koraku smo v drevesu objektov poiskali vsa imena ponudnikov in zadetke shranili v seznam *restaurantNames*. Imena smo v drevesu poiskali z jezikom XPath. Poizvedbo *//div[@class="name"]/h1/a/text()* si lahko razlagamo kot: "Vrni vrednost

elementa *a*, ki je otrok elementa *h1*, ki je otrok elementa *div* z atributom *class="name"*, ki se nahaja pri korenu drevesa."

```
from lxml import html
import requests

#Get page object from the URL
page = requests.get('https://www.studentska-prehrana.si/Pages/
    Directory.aspx')

#Form tree from the page elements
pageTree = html.fromstring(page.text)

#Create list of restaurant names using XPATH
restaurantNames = pageTree.xpath('//div[@class="name"]/h1/a/text
    ()')
```

Izsek kode 4.1: Primer kode za grabljenje podatkov.

V skripti smo uporabili tudi Geopy knjižnico in sicer Googlov API za pridobitev koordinat o ponudniku. Zaradi boljšega delovanja smo pridobili tudi svoj API ključ in poizvedbe omejili na vsake 1,5 sekunde. Skripta za pridobivanje podatkov se v grobem izvaja po naslednjih korakih:

1. Pridobi ime, podatke o cenah in povezavo do dodatnih informacij.
2. Uporabi povezavo o dodatnih informacijah za kreiranje novega iskalnega drevesa.
3. Iz novega drevesa pridobi podatke o naslovu podnudnika.
4. Uredi pridobljene podatke (rezanje nepotrebnih znakov, odstranitev presledkov ...).
5. Z uporabo geolokatorja in naslova pridobi koordinate ponudnika.
6. Generiraj SQL stavke in jih shrani v datoteko, primerno za uvoz v podatkovno bazo.

Sledila je še izdelava skripte za posodabljanje podatkov o ponudnikih. Da ne bi po nepotrebnem obremenjevali podatkovne baze, smo se odločili, da bi bila najboljša rešitev primerjava stare datoteke podatkov z novo. Koraki skripte za posodabljanje podatkov:

1. Uporabi skripto za pridobivanje podatkov.
2. Primerjaj nove podatke s podatki iz (stare) datoteke.
3. Če je potrebno, posodobi podatke o ponudniku; če ponudnik ne nudi več subvencij ga ustrezno označi (polje *feePrice*).
4. Generiraj SQL stavke in jih shrani v datoteko, primerno za uvoz v podatkovno bazo.

4.4 Izdelava spletne aplikacije

Izdelava spletne aplikacije se je delila na dva dela in sicer na izdelavo spletne strani in programiranje logike na strani strežnika. Spletna stran se je programirala v jeziku Jade in jQuery. Na strežniku smo programirali v jeziku JavaScript, ki se izvaja v okolju Node.js.

4.4.1 Izdelava spletne strani

Odločili smo se, da se bo naša stran delila na tri dele:

- glavno in hkrati tudi domačo stran, kjer so prikazani vsi ponudniki, iskalnik in možnosti za razvrščanje,
- stran ponudnika, kjer so prikazani vsi oddani komentarji in možnosti za razvrščanje komentarjev,
- stran s zemljevidom, kjer uporabnik lahko išče ponudnike v svoji neposredni bližini.

Podatke, kot so na primer podatki o ponudnikih pošiljamo, odjemalcu v obliki JSON [16] skupaj z odgovorom na HTTP zahtevek. Uporabimo jih lahko neposredno v Jade predlogi ali pa posredujemo skriptnim datotekam.

```
//...//
- each restaurant in results
  .panel.panel-primary.panel-restaurant(data-restaurant-id =
   restaurant.restaurant_id)
    - var restaurantLink = "/restaurants/"+restaurant.
    restaurant_id;
     a.info-page-link(href=restaurantLink)
       .panel-heading.panel-heading-index-restaurant
         span.panel-title #{restaurant['name']}
         if restaurant.averageRating == null
           span.panel-rating neocenjeno
         else
           span.panel-rating #{restaurant['averageRating']}/5
//...//
```

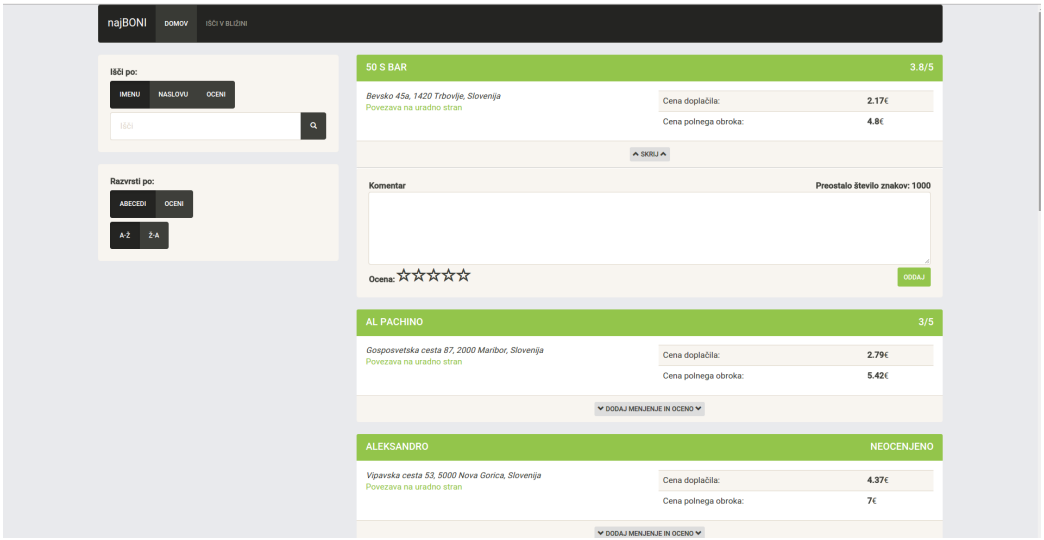
Izsek kode 4.2: Primer kode za generiranje glave elementa *'ponudnik'*.

Jade jezik v principu zelo podoben HTMLju. Nima začetnih in končnih značk. Hierarhijo elementov določamo z zamiki. Primer izseka programske kode 4.2 prikazuje generiranje glave elementa 'ponudnik'. *Results* so podatki, ki smo jih pridobili s HTTP zahtevkom. V kodi lahko uporabljamo tudi medvrstični JavaScript. Jade med drugim že v osnovi omogoča pogojne stavke, ki smo jih v dotičnem primeru uporabili za izpis ocene glede na to, ali je ponudnik že bil ocenjen ali ne.

4.4.1.1 Domača stran

Domačo stran, ki je prikazana na sliki 4.3, bi lahko razdelili na levi in desni stolpec. V levem stolpcu imamo polji za iskanje in razvrščanje ponudnikov. V desnem stolpcu pa prikazujemo ponudnike.

Uporabnik lahko išče ponudnike glede na ime, naslov ali oceno. Iskanje se



Slika 4.3: Domača stran spletne strani.

izvaja v realnem času, sproti, ko uporabnik vnaša besedilo. Pri iskanju z oceno, se vnosno polje zamenja z drsnikom. Tako smo na enostaven način preprečili napačne in neveljavne vnose.

Izsek kode 4.3 prikazuje implementacijo iskanja v realnem času (zaradi preglednosti se izsek osredotoča samo na primer, ko iščemo z imenom). Najprej pridobimo vse HTML elemente z atributom *class="panel-restaurant"*, ki predstavljajo naše ponudnike. Nato iz vnosnega polja pridobimo vnešeno vrednost. V naslednjem koraku nad poljem, ki vsebuje naše elemente ponudnikov, izvedemo zaporedje ukazov. Funkcija *show()* prikaže vse ponudnike, nato funkcija *filter()* nad prikazanimi zajame tiste, ki v imenu ne vsebujejo besedne zveze, ki jo iščemo, in nazadnje funkcija *hide()* skrije zajete elemente. Funkcija *searchResults()* se proži vsakič, ko nekaj vnesemo v iskalno polje.

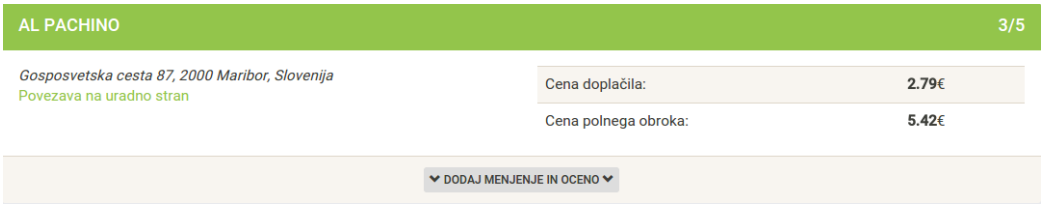
```
// Attach listener - Filter search results
$('#search-field').on('input', searchResults);

// Filter and show search results which match search string
function searchResults(){
```

```
    // Get all restaurants
    var restaurants = $('.panel-restaurant');

    //Get value from the search field
    var searchValue;
    // .... //
    // Get value from the input
    searchValue = $(this).val().toLowerCase();

    // Show only panels which contain words or letters in the
    search field (!~ means negate)
    restaurants.show().filter(function() {
        // Depending on the searchVariable get right text
        var text;
        // .... //
        // Search name
        text = $(this).find('.panel-title').text().toLowerCase()
    ;
        return !text.indexOf(searchValue);
    }).hide();
};
```

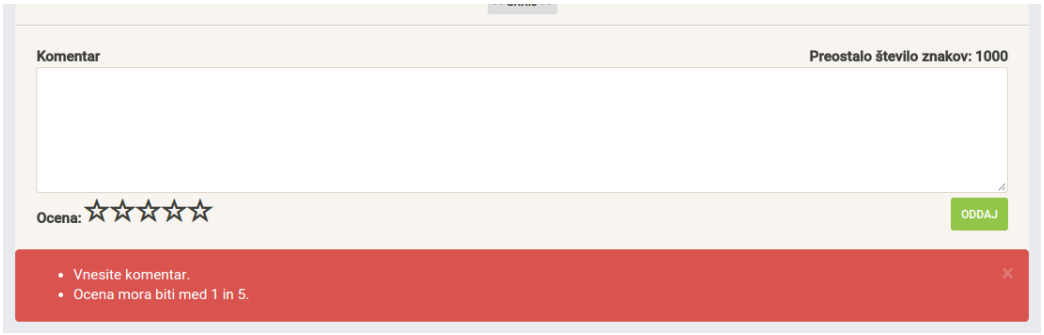
Izsek kode 4.3: JQuery koda za implementacijo iskanja ponudnikov v realnem času.

Uporabnik lahko razvršča ponudnike po abecednem vrstnem redu in glede na oceno. Razvrščanje se prav tako kot iskanje izvaja v realnem času. Najprej pridobimo seznam elementov. Elemente nato sortiramo s sortirno funkcijo glede na izbrani parameter. Stare elemente odstranimo iz HTML dokumenta in pripenjamo tiste, ki smo jih sortirali. Pripenjanje se iz seznama vrši zaporedno tako, da tudi v HTML dokumentu dosežemo željeno ureditev.

Vsakega ponudnika predstavimo s *ploščo (panel)*, ki je prikazana na sliki 4.4. V glavi plošče se prikažeta ime in ocena. Če ponudnik ni bil še nikoli ocenjen, se namesto ocene prikaže beseda *neocenjeno*. Glava služi tudi kot povezava do strani o ponudniku. V telesu plošče izpisujemo naslov, povezavo do po-



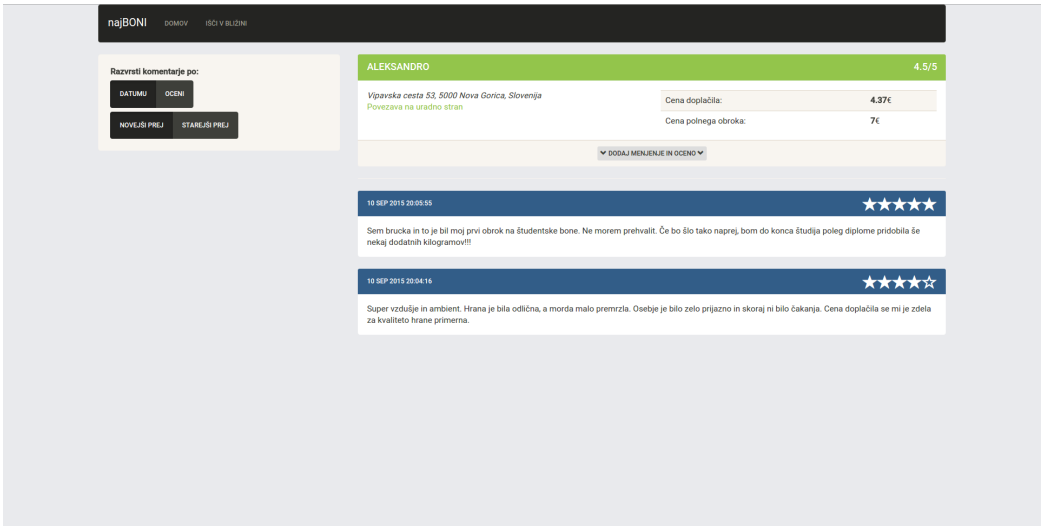
Slika 4.4: Plošča (panel) ponudnika.



Slika 4.5: Če ne vnesemo komentarja ali ocene, se nam prikaže opozorilno sporočilo.

nudnikove strani na uradni strani študentske prehrane, vrednost celotnega obroka in ceno doplačila. Povezavo do uradne strani smo vključili, ker na svoji strani nismo hoteli prikazovati preveč podatkov, ampak smo vseeno hoteli uporabniku ponuditi možnost hitrega dostopa do ostalih informacij, kot so menuji in delovni čas. V nogi plošče se nahaja gumb, s katerim odpremo okno za komentiranje in ocenjevanje. Uporabnik ima za komentar na voljo 1000 znakov. Preostalo število znakov se posodablja v realnem času. Oceno se izbere s klikom na poljubno število zvezdic.

Po kliku gumba *oddaj* se opravi še validacija vnosov. Validacijo vnosov preverjamo tako na strani odjemalca kot na strani strežnika. Za validacijo na strani klienta smo se odločili, da ne po nepotrebnem pošiljamo zahtevkov na strežnik, če so vnosi pomanjkljivi ali napačni. Slika 4.5 prikazuje opozorilno sporočilo, ki se pojavi pod oknom za komentiranje na primer, če ne vnesemo komentarja, ocene, ali obeh.



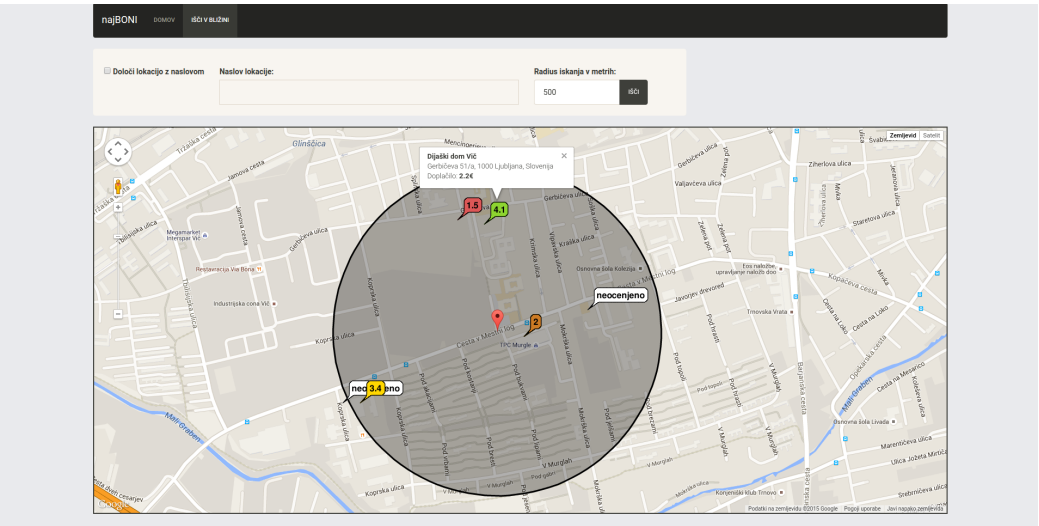
Slika 4.6: Stran ponudnika, kjer so prikazani tudi vsi komentarji.

4.4.1.2 Stran ponudnika

Stran ponudnika, ki je prikazana na sliki 4.6, lahko spet razdelimo na levi in desni stolpec. Levi stolpec vsebuje možnosti za razvrščanje komentarjev, desni pa vsebuje podatke o ponudniku z možnostjo dodajanja komentarja. Komentarje lahko razvrščamo glede na datum dodajanja ali glede na oceno. Dodajanje komentarja deluje enako kot dodajanje na glavni strani. Komentar in posodobljena ocena se prikažeta komaj ob ponovni osvežitvi strani.

4.4.1.3 Stran za iskanje ponudnikov v bližini

Iskanje ponudnikov v bližini trenutne lokacije je ena izmed glavnih funkcionalnosti naše aplikacije. Realizirali smo jo s pomočjo Googlovih zemljevidov. Ob nalaganju strani, poskuša aplikacija samodejno določiti položaj uporabnika z uporabo *navigator.geolocation* (v nadaljevanju geolokator) programskega vmesnika, ki je del HTML5 standarda. Geolokator lahko pridobi lokacijo preko modula GPS, omrežja Wi-Fi ali IP naslova. Ker geolokator ni vedno natančen ali pa uporabnik ne dovoli samodejnega določanja lokacije



Slika 4.7: Iskanje ponudnikov v bližini trenutne lokacije.

ali pa hoče iskati ponudnike na specifični lokaciji, na kateri se trenutno ne nahaja, smo dodali tudi možnost, da uporabnik sam vpiše naslov trenutne lokacije.

Uporabnik na zemljevidu prikaže ponudnike tako, da v polje *radius iskanja v metrih* vpiše vrednost, s katero določi, v kakšnem obsegu od trenutne lokacije bi rad iskal ponudnike. Po kliku gumba *išči*, se na zemljevidu označi iskalno območje in ponudniki, ki se nahajajo v njem. Za vsakega ponudnika, ki jih iz baze pridobimo ob nalaganju strani, se z uporabo koordinat in formule za določanje razdalje med dvema točkama na zemljini površini [10] izračuna razdalja med ponudnikom in lokacijo uporabnika. Če je razdalja enaka ali manjša razdalji, ki jo je vnesel uporabnik, se zgenerira nova označba z ustreznimi podatki, ki se nato pripne na zemljevid, kar je razvidno iz slike 4.7. Na vsaki oznaki je izpisana trenutna ocena ponudnika. Dodatno se oznaka glede na vrednost ocene tudi obarva, kar omogoča lažje razlikovanje. S klikom na oznako se nam odpre informativno okno, ki prikaže nekaj osnovnih informacij. S klikom na informativno okno dostopamo do strani ponudnika.

4.4.2 Strežniški del

Programiranje na strani strežnika se je v grobem delilo na:

- povezovanje modulov, povezovanje usmerjevalnikov, vzpostavitev povezave z bazo v datoteki *App.js*,
- programiranje usmerjevalnikov.

4.4.2.1 Datoteka App.js

Kot smo že omenili v podpoglavju 4.1, ogrodje Express.js zgenerira dve pomembni datoteki in sicer datoteko *www* in *App.js*. Datoteka *www* definira naslov in vrata na katerih poslušamo in kliče metodo, ki ustvari instanco strežnika. Za nas, z vidika programiranja je bila bolja zanimiva datoteka *App.js*, ki je prikazana v izseku kode 4.4.

```
var express = require('express');
var path = require('path');
//.../

var routes = require('./routes/index');
var findnear = require('./routes/findnear');
var restaurants = require('./routes/restaurants');

var app = express();

//Mysql and database
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'root',
  password : 'root',
  database : 'foodie'
});

//Make our mysql connection accessible to our router
```

```
app.use('/', function(req,res,next){
    req.connection = connection;
    next();
});

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

//Set path to public folder
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', routes);
app.use('/findnear', findnear);
app.use('/restaurants', restaurants);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handlers
//.../

module.exports = app;
```

Izsek kode 4.4: Okrnjena vsebina datoteke App.js.

Na začetku z uporabo metode *require()* uvozimo vse potrebne module in usmerjevalnike, na katere se lahko pozneje v kodi sklicujemo preko spremenljivk. Nato z metodo *express()* ustvarimo instanco *Express* aplikacije. V nadaljevanju uvozimo še modul za MySQL in definiramo povezavo do podatkovne baze. Ker bi radi povezavo do baze definirali le enkrat, jo najprej pri vsakem zahtevku HTTP, z uporabo metode *use()* pripnemo objektu zahtevka HTTP *req* in nato zahtevek z metodo *next()* damo na voljo ustreznemu

usmerjevalniku. V usmerjevalniku lahko nato iz objekta *req* pridobimo povezavo.

V nadaljevanju aplikaciji določimo še poti do map *views* in *public* ter določimo Jade kot privzeti pogon za prikazovanje predlog. V naslednjem koraku na enotne označevalnike virov, ki jih uporabljamo na spletni strani, pripnemo ustrezne usmerjevalnike. V datoteki *App.js* imamo definirano tudi logiko za odziv na napake, kot je recimo napaka *HTTP 404*.

4.4.2.2 Usmerjevalniki

Namen usmerjevalnikov je, da prestrezajo zahtevke HTTP, ki jih neposredno ali posredno pošilja uporabnik. Usmerjevalnik se izbere glede na enotni označevalnik vira (URI). Na primer, če dostopamo do strani `www.najboni.si/restaurant/13`, se bo zahtevek HTTP preusmeril do usmerjevalnika *router.get('/restaurant/:id', function(req, res, next)*, kjer se nanj ustrezno odzvamo.

Primer usmerjevalnika, ki pri naši aplikaciji skrbi za prikaz domače strani, je prikazan v izseku kode 4.5. Če pride do zahtevka GET iz korena spletnega naslova, najprej pridobimo povezavo do podatkovne baze. Nato opravimo poizvedbo, s katero pridobimo vse ponudnike. Če pri poizvedbi ni prišlo do napake, kot odgovor na zahtevek HTTP posredujemo Jade predlogo *index* in poleg pošljemo še objekt *results*, ki vsebuje podatke o ponudnikih.

```
var express = require('express');
var router = express.Router();

// GET home page/index.
router.get('/', function(req, res, next) {
  var connection = req.connection;
  connection.query('SELECT * FROM Restaurants WHERE feePrice !=
   -1 ORDER BY name', function(error, results, fields){
    if(!error){
      res.render('index', {results : results, title : 'Foodie'})
```

```
    }else{
      console.log("Error @query execution: "+error);
    }
  });
});
```

Izsek kode 4.5: Primer usmerjevalnika za prikaz domače strani.

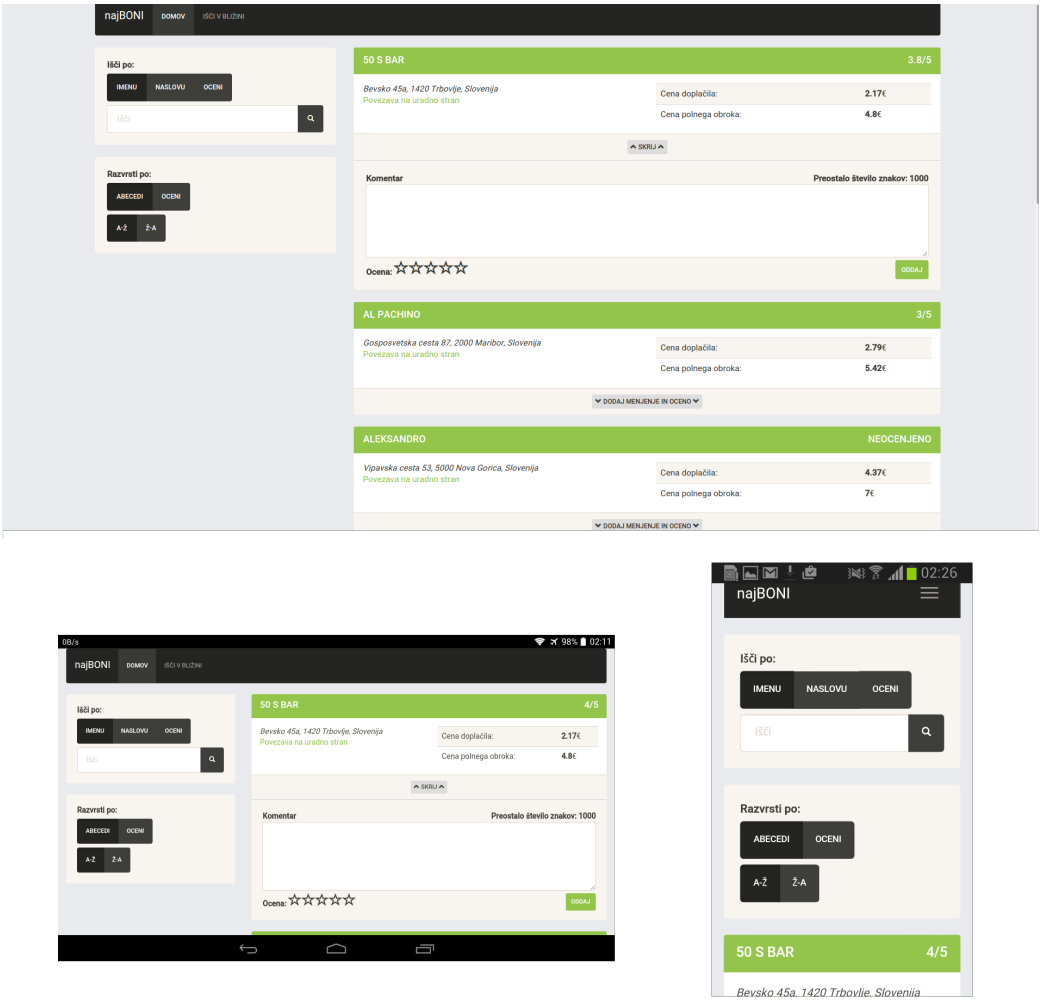
4.4.3 Odzivnost spletne strani (RWD)

Spletne strani so navadno razvite za uporabo na zaslonih z večjimi resolucijami. Posledično se strani na manjših zaslonih, kot so na primer zasloni prenosnih naprav, večkrat ne prikazujejo pravilno. Ker smo hoteli zagotoviti uporabnost naše aplikacije tudi na pametnih telefonih in tabličnih računalnikih, smo našo spletno stran optimizirali, da se pravilno, oziroma optimalno prikazuje tudi pri manjših resolucijah.

Pri implementaciji odzivnosti nam je zelo pomagalo ogrodje Bootstrap. Bootstrap v osnovi razdeli stran na 12 stolpcev in tako tvori mrežni sistem *(grid system)*. Elemente razporejamo po mreži s štirimi skupinami razrednih atributov:

- razred *col-xs-**, je namenjen resolucijam, ki so v širino manjše od 768 pikslov,
- razred *col-sm-**, je namenjen resolucijam, ki so v širino večje ali enake 768 pikslov,
- razred *col-md-**, je namenjen resolucijam, ki so v širino večje ali enake 970 pikslov,
- razred *col-lg-**, je namenjen resolucijam, ki so v širino večje ali enake kot 1200 pikslov.

V primeru naše domače strani smo uporabili kombinacijo razredov, ki je prikazana v izseku kode 4.6. V primeru velikih računalniških zaslonov *(col-*



Slika 4.8: Prikaz spletne strani na različnih napravah. Zgoraj računalnik, spodaj levo tablični računalnik in spodaj desno mobilni telefon.

lg-)*, se najprej odmaknemo za eno enoto (stolpec) v desno, plošči za iskanje in sortiranje raztegnemo čez tri enote in plošče ponudnikov raztegnemo čez sedem enot. Ker se elementi nahajajo v isti vrsti *(row)*, dokument napolnijo horizontalno zaporedno in tako zavzamejo vseh 12 osnovnih stolpcev *(1 + 3 + 7 (+ 1) = 12)*.

```
.container-fluid
  .row
    .col-lg-offset-1.col-lg-3.col-md-4.col-sm-4
      // ... Ploscaa za iskanje ... //
      // ... Plosca za sortiranje ... //

    .col-lg-7.col-md-8.col-sm-8
      // ... Plosce ponudnikov ... //
```

Izsek kode 4.6: Definicija razrednih atributov za doseganje odzivnosti domače strani.

Pri razporeditvi elementov na zelo majhnih zaslonih nam pomaga razred *container-fluid*, ki sam optimalno razporedi elemente in grupira navigacijsko vrstico. Prikaz strani na različnih napravah prikazuje slika 4.8.

4.4.4 Opis končne aplikacije

Končna aplikacija omogoča iskanje ponudnikov v realnem času glede na ime, naslov ali oceno. Rezultate iskanja lahko razvrščamo po abecednemu redu ali pa glede na vrednost ocene. Vsakemu ponudniku lahko dodamo opisni komentar dolžine največ 1000 znakov in oceno med 1 in 5. Oddane komentarje si lahko za vsakega ponudnika posebej ogledamo na njegovi strani, kjer jih lahko razvrščamo glede na datum dodajanja ali oceno. Dodatno, lahko ponudnike iščemo v bližini trenutne lokacije, ki se privzeto določi sama, dodana pa je tudi možnost, da jo določi uporabnik sam. Morebitni ponudniki, ki so rezultat iskanja, se na zemljevidu prikažejo kot oznake s trenutnimi ocenami, preko katerih lahko dostopamo do dodatnih informacij.

Poglavje 5

Sklep

V okviru diplomske naloge smo razvili spletno aplikacijo, ki uporabnikom omogoča ocenjevanje, komentiranje in iskanje ponudnikov prehrane na študentske bone. Prestavili smo uporabljene tehnologije in orodja. Podrobneje smo opisali potek vzpostavitve delovnega okolja, načrtovanje podatkovne baze in izdelavo skript za pridobivanje in posodabljanje ponudnikov. Poudarek smo dali tudi na opis izdelave spletne strani in kode na strani strežnika.

Ugotovili smo, da je pri razvoju spletnih aplikacij zelo pomembna dobra definicija zahtevanih funkcionalnosti, da je razvoj bolj voden in nadzarovan. Morda bi bilo dobro, da bi aplikacijo razvijali po principu kakšne izmed metod za razvijanje programske opreme, kot so na primer agilne metode razvoja programske opreme [1].

Med izdelavo se nam je porodilo mnogo idej, kako bi lahko aplikacijo izboljšali in nagradili. Ene izmed možnih izboljšav so:

- izdelava namenske mobilne verzije spletne strani, na primer z uporabo ogrodja jQuery Mobile [15],
- integracija aplikacije z družabnimi omrežji,
- razviti sistem, ki bi uporabniku predlagal zanj primerne ponudnike.

Kljub temu, da je še veliko možnosti za izboljšave, smo mnenja, da je izdelana spletna aplikacija v okviru diplomskega dela realizirana v skladu s zahtevami in začetnimi pričakovanji.

Literatura

[1] Agile software development. Dostopano, Avgust 2015. https://en.wikipedia.org/wiki/Agile_software_development.

[2] Ajax. Dostopano, Avgust 2015. https://en.wikipedia.org/wiki/Ajax_(programming).

[3] Bootstrap. Dostopano, Avgust 2015. http://getbootstrap.com/.

[4] Bootstrap rating. Dostopano, Avgust 2015. https://github.com/dreyescat/bootstrap-rating.

[5] Bootswatch template. Dostopano, Avgust 2015. https://bootswatch.com/sandstone/.

[6] Css. Dostopano, Avgust 2015. https://en.wikipedia.org/wiki/Cascading_Style_Sheets.

[7] Express.js. Dostopano, Avgust 2015. http://expressjs.com/.

[8] Geopy. Dostopano, Avgust 2015. https://geopy.readthedocs.org/en/1.10.0.

[9] Google maps api. Dostopano, Avgust 2015. https://developers.google.com/maps/.

[10] Haversine formula. Dostopano, Avgust 2015. https://en.wikipedia.org/wiki/Haversine_formula.

[11] Html. Dostopano, Avgust 2015. http://www.w3.org/html.

[12] Imenik študentske prehrane. Dostopano, Avgust 2015. https://www.studentska-prehrana.si/Pages/Directory.aspx.

[13] Jade. Dostopano, Avgust 2015. http://jade-lang.com/.

[14] Jquery. Dostopano, Avgust 2015. https://jquery.com/.

[15] Jquery mobile. Dostopano, Avgust 2015. https://jquerymobile.com.

[16] Json. Dostopano, Avgust 2015. http://json.org.

[17] Lxml. Dostopano, Avgust 2015. http://lxml.de.

[18] Mixin. Dostopano, Avgust 2015. https://en.wikipedia.org/wiki/Mixin.

[19] Mongodb. Dostopano, Avgust 2015. https://www.mongodb.org/.

[20] Mysql. Dostopano, Avgust 2015. https://www.mysql.com/.

[21] Node.js. Dostopano, Avgust 2015. https://nodejs.org/.

[22] Nosql. Dostopano, Avgust 2015. https://en.wikipedia.org/wiki/NoSQL.

[23] phpmyadmin. Dostopano, Avgust 2015. https://www.phpmyadmin.net/.

[24] Python. Dostopano, Avgust 2015. https://www.python.org/.

[25] Sql. Dostopano, Avgust 2015. https://en.wikipedia.org/wiki/SQL.

[26] Sublime text. Dostopano, Avgust 2015. http://www.sublimetext.com/.

[27] Xpath. Dostopano, Avgust 2015. http://www.w3.org/TR/xpath.